UNIT I – FUNDAMENTALS

Principles of Testing – Phases of Software Project – Quality Assurance and Control – Verification and Validation - White Box Testing: Static Testing – Structural Testing – Challenges.

1.Software Testing:

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest.

The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. In simply, Software Testing means the Verification of Application Under Test (AUT). This tutorial introduces testing software to the audience and justifies its importance.

Software testing is an important part of the software development lifecycle that involves verifying and validating whether a software application works as expected. It ensures reliable, correct, secure, and high-performing software across web, mobile applications, cloud, and CI/CD pipelines in DevOps.

- Detect bugs before release.
- Ensure software meets requirements.
- Improve product quality and reliability.
- Reduce cost of fixing issues later.
- Validate performance, security, and usability.

The following professionals are involved in testing a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality, Assurance Engineer, QAAnalyst, etc.

A test lead is responsible for:

- Defining the testing activities for subordinates – testers or test engineers.
- All responsibilities of test planning.
- To check if the team has all the necessary resources to execute the testing activities.
- To check if testing is going hand in hand with the software development in all phases.

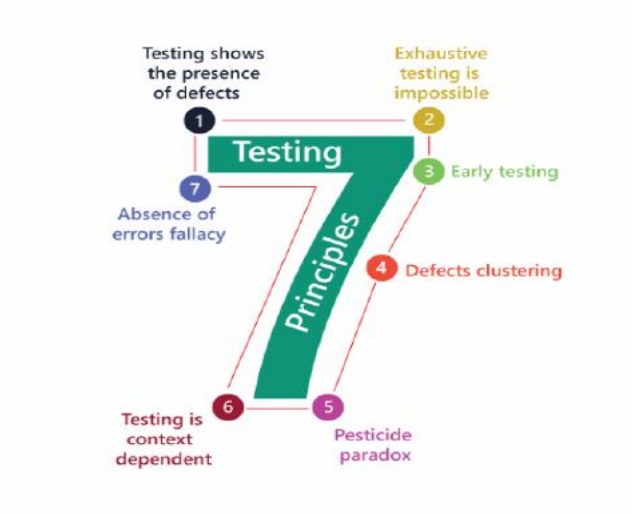
- Prepare the status report of testing activities.
- Required Interactions with customers.
- Updating project manager regularly about the progress of testing activities.

Test engineers/QA testers/QC testers are responsible for:

- To read all the documents and understand what needs to be tested.
- Based on the information procured in the above step decide how it is to be tested.
- Inform the test lead about what all resources will be required for software testing.
- Develop test cases and prioritize testing activities.
- Execute all the test case and report defects, define severity and priority for each defect.
- Carry out regression testing every time when changes are made to the code to fix defects.
- Testing is done in different forms at every phase of SDLC −
- During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
- Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
- Testing performed by a developer on completion of the code is also categorized as testing.

1.1 Principles of Software Testing

- Testing shows the presence of defects
- Exhaustive Testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context-dependent
- Absence of errors fallacy



- All the test cases should be able to satisfy the end users specifications.
- Testing is performed to detect defects in the software.
- Exhaustive testing is not possible.
- Defect clustering which means maximum count of defects are detected from a small number of modules.
- Pesticide Paradox which means that the same number of test cases if used repeatedly does not detect new defects in the software.
- The testing activities should be kicked off from the early stages of the software development life cycle (SDLC).
- Testing depends on the context.
- Testing is not only concerned with finding defects but also to confirm that the software has been built as per the user requirements.

1. Testing Shows Presence of Defects

- Tests may demonstrate that defects are there; they can never demonstrate that there are no defects. This means that while testing can show problems, it cannot ensure that any software is entirely free from defects.
- Testing serves to find flaws within the software so that at least some improvement occurs before its release.

2. Exhaustive Testing is Impossible

- Testing every possible scenario, situation, and course is impractical and not possible due to a couple of diversifications and combinations. Rather, testers ought to goal on threat analysis and prioritize trying out efforts to cover the most important and impactful areas of the software. With the aid of identifying and focusing on the maximum susceptible parts, testers can maximize their performance and effectiveness.

3. Early Testing

- Testing should begin as early as possible in the software development lifecycle. The shift-left approach provides an atmosphere for early defect detection as opposed to late defect detection, which is less expensive and easier to fix.
- By integrating testing from day one, teams can detect problems early and reduce the cost and effort associated with rectifying those problems.

4. Defect Clustering

- Experience suggests that a small, wide variety of modules frequently includes the maximum defects. This phenomenon, called illness clustering, shows that defects are not frivolously distributed across the software program.
- By figuring out those high-risk regions, testers can be aware of their efforts at the parts of the software program that are most likely to incorporate defects, thereby enhancing the effectiveness of their testing.

5. Pesticide Paradox

- In time, performing the same tests will render them useless as they will cease to discover new defects. This is called the Pesticide Paradox. Hence, it becomes mandatory to regularly reevaluate and modify the test cases.
- New and different tests should be written to exercise various parts of the software. This approach ensures that testing remains effective and continues to uncover new defects.

6. Testing is Context Dependent

- Testing approaches, techniques, and tools should be chosen based on the specific context of the software. For instance, safety-critical software programs, such as clinical gadgets or aviation systems, call for distinct testing approaches as compared to a simple e-trade website.
- Expertise in the context facilitates testers practice the maximum suitable strategies to ensure the software meets its intended use and fine requirements.

7. Absence-of-Errors Fallacy

- Although no defects are found, it doesn’t imply the software program is beneficial or meets user requirements. The absence-of-mistakes fallacy highlights that locating and fixing defects isn't always enough.
- The software program must also satisfy the user’s requirements and offer value. It’s crucial to make certain that the software is usable, practical and meets the expectations of its users.